8-15-1997

ETHICAL RESPONSIBILITIES OF THE SOFTWARE-DEPENDENT ORGANIZATION

Judith Pinn Carlisle
Georgia Institute of Technology, judith.carlisle@mgt.gatech.edu

ETHICAL RESPONSIBILITIES OF THE SOFTWARE-DEPENDENT ORGANIZATION

Judith Pinn Carlisle
DuPree School of Management
Georgia Institute of Technology
Atlanta, GA 30332-0520
(404) 894-4365
judith.carlisle@mgt.gatech.edu

Abstract

As organizations become more dependent upon software-based information systems, the potential for unforeseen crises increases. In this paper, we explore the relationship between software failure and organizational responsibilities.

0.0 Introduction

The author of this short essay once received the following comments from a reviewer:

`` (in this paper) a great deal of time is spent stating something quite obvious: that IR information retrieval) systems should be useful and reliable."

While the blithe observation is quite true, reality reflects a different situation. Information systems are not reliable, and so cannot be considered useful. Users and designers of information systems seem to forget that IS are not only computer-based systems, they are software-bound systems. Hence, these systems experience the same sorts of failures and vulnerabilities which characterize software.

In this short essay, the consequences of organizational dependence upon software-based technologies is explored. Toward this end, a historical basis for the examination of ``computer ethics" is established. Then, the need to include special issues regarding software into ethical thought about automation and the organization is proposed in order to explore the following premise: an organization has a responsibility to ensure its survival, internal health, and growth toward the goal of accomplishing its mission. Finally, an answer is constructed to the following question. What are the responsibilities which an organization takes onto itself which result from reliance upon software-bound information technologies?

1.0 Frameworks for Computer Ethics

The works of two authors [Moor], [Mason] have greatly influenced exploration and discussion of computer ethics. Mason's PAPA model identifies the four ``basic rights" which a society has with regard to information technologies. These are the right to:

- privacy,

- accuracy (of information),

- property, and

- access (to information).

[Moor] identifies factors unique to information technologies which must be considered when exploring how automation affects the ethical beliefs of a society. These factors are:

- logical malleability of software,

- transformation ability of software,

- the ``invisible" factors:

invisible programming factors,

invisible complex calculations, and

invisible abuse.

Mason's work examines ways in which information technologies affect these basic rights which a society has, while Moor's work identifies unique aspects of information technologies which result in changes to how we think about these rights. Both of these influential works identify consequences of the results of automation. Still, neither considers in detail how the underlying basis of automation --- software --- affects ethical considerations of a society, organization, or individual.

A great deal of thought and experimentation has been undertaken in order to understand what software is and to ascertain its effect on systems created using software. As a result of this work, a very strong conclusion about software can be made. This is that the complex nature of software results in inability to verify its correctness. That is, complex software systems cannot be shown to be reliable [Leveson and Turner], [Borning], [Parnas]. While we also know that complex software systems have failed in the past, with dire consequences, methods to build safer, more reliable software still fall short [Leveson], [Leveson and Turner], [Littlewood and Strigini].

These considerations lead to the observation that organizations which rely upon software-based information technologies to manage their information resource are reliant upon high risk systems. Organizations are dependent on systems which are neither reliable, nor correct.

2.0 On the Nature of Software

In this section, special considerations of software not commonly pursued in the management literature are introduced. Software is a logical product which uses discrete functions to achieve some desired effect. A number of authors have identified a number of factors which are unique to software. [Parnas] illustrates that fundamental technological differences exist between software products and other products which have been successfully produced. Influences contributing to the unreliable nature of software include, but are not limited to, human lack of understanding of discrete functions and poor software development practices. Poor software development practices result in poorly designed software being released, software being released, not when its reliability can be demonstrated, but after the rate of ``bug'' discovery falls below an acceptable threshold [Littlewood and Strigini]. Corbato recognizes the following contributing factors. Complexities arising from:

- the large scale of most software products,

- the many levels of management involved in development of software products,

- subordinate reluctance to report bad news,

- the youth of software development,

-rapid changes occurring in the technological realm.

Software weaknesses cannot be overcome or averted by attempts to correct failures as they are detected:

``...Most assume the failure rate is reduced whenever an error is corrected. They also assume the reduction in failure rates resulting from each correction are predictable. These assumptions are not justified by either theoretical or empirical studies of programs." [Parnas, van Schouwen and Kwan, p. 447]

Much of the work cited deals with large software systems, frequently these systems control and/or manage complex processes. Many organizational information systems are of a smaller nature and their failure will not lead to extreme crises such as the Challenger or Therac-25 disasters. However, software related failures of these systems can result in major crises within an organization.

3.0 Software and the Organization

The management of, and interaction with, no other resource is handled as carelessly as the information resource. Software-dependent organizations pursue a conscious policy of software-oriented ignorance, evading the effort of responsibly integrating and managing software development effort. Management of software and system development is then unselfconsciously relegated to the control of a small number of information technology specialists. The nature of software, and development of software-based applications, contributes greatly management's lack of understanding of the information resource. The

ability to understand and interact with software is still much relegated to the realm of the ``nerd." The organization has abdicated control over the information resource management process in an unprecedented manner. This leads to:

- lack of non-information technology

specialist influence over the software

development process,

- lack of non-information technology specialist understanding of vulnerabilities and risks of the IS,

- organizations allowing more and more complex tasks to be automated without adequate preparation for possible failure of these systems,

- lack of development of useful measures which will allow the organization to remove information resource management from the hands of a small cadre of individuals,

- development of methods which allow information resource management to be audited and controlled.

The organization which uses software applications can either develop software in-house, or rely on software developed by another organization. Although organizations, for the most part, now understand that information is a critical resource and have included a strategic planning function for

information resource management, actual control of the software development process remains an alien undertaking.

Historically, the software dependent organization engages in a naive relationship with regard to the IS development function, relegating IS implementation and maintenance to a relatively small number of highly skilled and knowledgeable professionals. The organization then proceeds to depend upon the ISes developed by these professionals with little or no questions as to the system's inherent reliability and safety. Conversely, and ironically, it is lack of understanding of organizations units which do not participate in the software development process which perpetuates and aggravates the notion of ``gutless estimating," [Brooks] that is, dismissing and overriding the time and costs estimates of software developing professionals without understanding of the methods and constraints which led to original estimations.

4.0 Conclusion

Despite the risks to which software dependence makes an organization vulnerable, organizations continue to allow the IS development function to remain isolated from other function areas of the organization. Organizations also maintain and re-enforce a

policy of willful ignorance regarding the software-based systems used to manage the information resource. Hence, the organization willfully acts in a manner which fails to ensures its own health and survivability, a manner which in fact, increases the possibility of crisis and decline. The organizations fails to fulfill the responsibilities it has with regard to understanding the consequences of software dependence. The dubious statement can be made then, that the organization willfully participates in activities which can lead to its own demise.

5.0 Citations

Borning, Alan. ``Computer system reliability and nuclear war,'' in Johnson, Deborah G., and Nissenbaum, Helen (eds). Computer Ethics and Social Values, Prentice-Hall, Englewood Cliff, NJ, 1995, pp. 398-421.

Brocklehurst, Sarah and Littlewood, Bev. ``New ways to get accurate reliability measures,'' IEEE Software, Vol. 9, July, 1992, pp. 34-43.

Brooks, Frederick P. Jr. The Mythical Man-Month, essays of software engineering, Anniversary Edition, Addison-Wesley, Reading, MA, 1995.

Corbato, Fernando J., ``On building systems that will fail,'' in Johnson, Deborah G., and Nissenbaum, Helen, (eds.), Computer Ethics and Social Values, Prentice-Hall, Englewood Cliff, NJ, 1995, pp. 421-437.

Jones, Capers. Patterns of Software Systems Failure and Success, International Thomson Computer Press, London, England, 1996.

Leveson, Nancy G. (JUDY BOOK) Safeware: system safety and computers, Addison-Wesley Publishing Company, Reading, MA, 1995.

Leveson, Nancy G., and Turner, Clark S. ``CASE: An investigation of the Therac-25 Accidents,'' in Johnson, Deborah G., and Nissenbaum, Helen, (eds.), Computer Ethics and Social Values pp. 474-513.

Littlewood, Bev, and Strigini, Lorenzo. ``The risks of software,'' in Johnson, Deborah G., and Nissenbaum, Helen, (eds.) *Computer Ethics and Social Values*, pp. 432-437.

Mason, Richard O. ``Four Ethical Issues of the Information Age,'' MIS Quarterly, Vol. 10, March, 1986, pp. 5-12.

McLeod, Raymond Jr. *Management Information Systems, A study of computer-based information systems*. Prentice-Hall, Englewood Cliffs, NJ, 1995.

Moor, James. ``What is Computer Ethics?'' Metaphilosophy, Vol. 16, October, 1985, pp. 266-275.

Smith, Brian Cantwell. ``Limits of correctness in computers,'''' in *Computer Ethics and Social Values*, Johnson, Deborah G., and Nissenbaum, Helen. (eds), pp. 456-467

Parnas, David L., van Schouwen A. John, Kwan, Shu Po. ``Evaluation of safety-critical software,'' Communications of the ACM, Vol. 33, June, 1990, pp. 636-649.

Parnas, David Lorge. ``Software Aspects of Strategic Defense Systems,'' American Scientist, September-October 1985, pp. 432-440.